# 《离散数学》课程实验报告6 Warshall

# 题目简介；

# 编写一个程序，实现Warshall算法用于计算给定关系矩阵的传递闭包。

# 二.解题思路；：

# 1. 输入矩阵大小：首先要求用户输入关系矩阵的大小n。

# 2. 输入关系矩阵元素：创建一个二维布尔类型的矩阵，允许用户输入n×n的关系矩阵元素，通常用0和1表示不同的关系。

# 3. 应用Warshall算法：实现Warshall算法的函数，根据传入的关系矩阵和大小n，计算传递闭包矩阵。Warshall算法的核心是通过中间节点k，不断更新矩阵中的元素，直到得到传递闭包

# 4. 输出传递闭包矩阵：在主函数中调用Warshall算法函数，得到传递闭包矩阵后输出结果，展示计算后的传递闭包矩阵。

# 5. 完善用户交互：在每个步骤中，通过输出提示信息和适当的说明，确保用户清楚地知道程序的运行状态和如何进行输入。

# 三.数据结构；

# 1. 二维布尔类型矩阵：用于表示关系矩阵和传递闭包矩阵。这个矩阵的行数和列数都是n，代表n个元素之间的关系。

# 2. 向量（Vector）：通过向量的嵌套使用，可以方便地表示二维布尔类型矩阵，并且能够动态地根据用户输入调整大小。

# 3. 整数（int）：用于表示矩阵的大小n，以及在算法中的循环和索引。

# 四. 实验原理和实现过程（算法描述）；

## 1.实验原理

Warshall算法的原理是通过迭代更新矩阵的元素，以计算传递闭包。其基本思想可以总结如下：

1. 初始化：首先，我们有一个布尔类型的邻接矩阵，代表了图中顶点之间的直接关系。矩阵中的每个元素matrix[i][j]表示顶点i到顶点j是否有直接边。

2. 传递闭包的计算：Warshall算法通过三重循环进行计算。在每一轮循环中，我们选择一个中间顶点k，并检查每一对顶点i和j之间的关系。如果顶点i和j之间没有直接边（matrix[i][j]为false），但是通过中间顶点k可以建立起连接（matrix[i][k]为true且matrix[k][j]为true），那么就更新matrix[i][j]为true，表示顶点i到顶点j存在一条路径。

3. 迭代更新：通过不断地选择不同的中间顶点k，更新矩阵中的元素，直到矩阵不再发生变化为止。在经过n次迭代后，我们就可以得到最终的传递闭包矩阵。

4. 输出结果：最终得到的矩阵即为传递闭包矩阵，其中的元素表示了图中顶点之间的传递关系。

## 通过这种方式，Warshall算法能够高效地计算出传递闭包，同时适用于有向图和无向图。这种基于动态规划的算法思想，使得它在实际应用中具有较高的效率和通用性。通过这种方式，Warshall算法能够高效地计算出传递闭包，同时适用于有向图和无向图。这种基于动态规划的算法思想，使得它在实际应用中具有较高的效率和通用性。

## 2.实验过程

# 1. 初始化：首先，我们有一个布尔类型的邻接矩阵，代表了图中顶点之间的直接关系。这个邻接矩阵是一个n×n的布尔类型二维矩阵，其中n表示顶点的个数。初始时，这个邻接矩阵中的元素matrix[i][j]表示顶点i到顶点j是否有直接边。

# 2. 传递闭包的计算：Warshall算法通过三重循环进行计算。外层循环选择中间顶点k（从0到n-1），中间层循环遍历矩阵的行i，内层循环遍历矩阵的列j。

# - 在每一次迭代中，检查当前矩阵元素matrix[i][j]。如果顶点i和j之间没有直接边（matrix[i][j]为false），但是通过中间顶点k可以建立起连接（matrix[i][k]为true且matrix[k][j]为true），那么就更新matrix[i][j]为true，表示顶点i到顶点j存在一条路径。

# 3. 迭代更新：重复进行三重循环的迭代，直到矩阵不再发生变化为止。在每次迭代中，我们都会更新矩阵中的元素，直到整个矩阵都不再发生变化为止。

# 4. 输出结果：最终得到的矩阵即为传递闭包矩阵，其中的元素表示了图中顶点之间的传递关系。

# 五.部分核心代码；

#include <iostream>

#include <vector>

using namespace std;

// 定义Warshall算法函数

void warshall(vector<vector<bool>>& matrix, int n)

{

for (int k = 0; k < n; k++) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

matrix[i][j] = matrix[i][j] || (matrix[i][k] && matrix[k][j]);

}

}

}

}

// 输入关系矩阵

void inputMatrix(vector<vector<bool>>& matrix, int n)

{

cout << "请输入" << n << "x" << n << "的关系矩阵（用0和1表示）：" << endl;

for (int i = 0; i < n; i++) {

vector<bool> row;

for (int j = 0; j < n; j++) {

bool value;

cin >> value;

row.push\_back(value);

}

matrix.push\_back(row);

}

}

int main()

{

int n;

cout << "请输入矩阵的大小：";

cin >> n;

// 定义关系矩阵

vector<vector<bool>> matrix;

// 输入关系矩阵

inputMatrix(matrix, n);

// 调用Warshall算法函数

warshall(matrix, n);

// 打印传递闭包矩阵

cout << "传递闭包矩阵：" << endl;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}